

CLAIMS

What is claimed is:

1 1. A policy compiler comprising:
2 a system controller to generate a configuration data abstraction layer of a
3 routing policy, the configuration data abstraction layer to map configuration to an
4 intermediate layer comprising fields, operators and arguments; and
5 a policy repository to verify the intermediate layer against a set of
6 verification rules for one or more client protocols including versions thereof, the
7 policy repository to generate compiled policy transmission language for use by the
8 one or more client protocols including versions thereof.

1 2. The compiler of claim 1 wherein the policy repository is to verify the
2 intermediate layer against the set of verification rules for a particular one or more
3 of a plurality of client protocols including versions of the one or more of the
4 plurality of client protocols, and
5 wherein the policy repository is to generate the compiled policy
6 transmission language for use by the one or more client protocols including
7 versions thereof.

1 3. The compiler of claim 2 wherein, for each statement in the policy, the
2 policy repository is to verify fields for each of the client protocols including
3 versions thereof, is to verify field-operator pairings in the policy, and is to verify
4 one or more arguments used for each field-operator pairing in the policy,
5 wherein the verifying is based on verification rules associated with a client
6 dynamic link library (DLL) for each of the client protocols including versions
7 thereof.

1 4. The compiler of claim 3 wherein the policy repository is to validate the
2 policy against at least two versions of a border gateway protocol (BGP), and
3 wherein the compiled policy transmission language is to be executed by
4 the at least two versions of the border gateway protocol.

1 5. The compiler of claim 3 wherein the policy repository is to further
2 validate the policy against one or more versions of a border gateway protocol
3 (BGP) client protocol, one or more versions of an open-shortest-path-first (OSPF)
4 client protocol and one or more versions of an intermediate system to intermediate
5 system (IS-IS) client protocol, and
6 wherein the compiler further comprises an execution engine associated
7 with each of the one or more versions of the client protocols, the associated
8 execution engines to separately execute the compiled policy transmission
9 language of the associated client protocol.

1 6. The compiler of claim 3 wherein the policy repository is to perform a
2 verification in response to a request for use of the policy for an attach point, the
3 attach point being a set of capabilities associated with a version of one of the
4 client protocols.

1 7. The compiler of claim 1 wherein the compiled policy transmission
2 language comprises a set of rules for each policy statement of a verified policy for
3 execution by an execution engine associated with ones of the client protocols and
4 versions thereof when verified against ones of the client protocols and versions
5 thereof.

1 8. The compiler of claim 1 further comprising an execution engine for each
2 of a plurality of client protocols and versions thereof,
3 wherein the execution engines are to execute the compiled policy
4 transmission language based on execution rules provided by an associated client
5 dynamic link library (DLL).

1 9. The compiler of claim 8 wherein the execution rules provided by one of
2 the client DLLs are used by an execution engine of the associated version of one
3 of the client protocols during execution of the compiled policy.

1 10. The compiler of claim 8 wherein the system controller is to perform an
2 optimization procedure to improve execution performance of the compiled policy
3 transmission language,
4 wherein as part of the optimization procedure comprises rearranging policy
5 statements to improve the execution performance,
6 wherein for a field-operator pair in an associated one of the client DLLs,
7 the system controller is to determine when a statement is processing intensive, and
8 is to rearrange at least some of the statements to check at least some processing
9 intensive during execution after less processing intensive statements.

1 11. The compiler of claim 10 wherein as part of the optimization
2 procedure, the system controller is to further eliminate at least some repeated
3 statements or repeated portions of statements, and
4 wherein the policy repository is to generate an optimized compiled policy
5 in policy transmission language prior to compiling.

1 12. The compiler of claim 1 wherein at least one of the versions of the
2 client protocols is to mark route attributes in a batch of routes as invariant for
3 route attributes that share values across differing route prefixes, and
4 wherein an execution engine associated with one of the client protocols is
5 to cache results for statements associated with the marked attributes when
6 executing the compiled policy transmission language,
7 wherein upon continued execution of the compiled policy transmission
8 language, the execution engine is to use the cached results for evaluations of
9 subsequent statements in the policy which reference the marked attributes.

1 13. A method of generating compiled policy for execution by one or more
2 client protocols including one or more versions of the client protocols, the method
3 comprising:
4 generating a configuration data abstraction layer for a routing policy to
5 map configuration to an intermediate layer;
6 verifying the intermediate layer against a set of rules for the one or more
7 client protocols including the one or more versions thereof; and
8 generating compiled policy transmission language when statements of the
9 intermediate layer are verified against the set of rules for the one or more client
10 protocols including the one or more versions thereof.

1 14. The method of claim 13 further comprising executing the compiled
2 policy transmission language with an execution engine associated with the one or
3 more client protocols including the one or more versions thereof, the executing
4 being based on execution rules for the one or more client protocols including the
5 one or more versions thereof, the execution rules being provided by a client
6 dynamic link library (DLL) associated with the one or more client protocols
7 including the one or more versions thereof.

1 15. The method of claim 14 wherein validating comprises verifying the
2 routing policy against a border gateway protocol (BGP) client protocol, an open-
3 shortest-path-first (OSPF) client protocol and an intermediate system to
4 intermediate system (IS-IS) client protocol, and
5 wherein executing further comprises separately executing, with an
6 execution engine associated with each of the client protocols including versions
7 thereof, the compiled policy transmission language.

1 16. The method of claim 13 wherein generating compiled policy
2 transmission language comprises performing an optimization procedure to
3 improved execution performance of the compiled policy transmission language,
4 wherein the optimization procedure comprises rearranging policy
5 statements to improve the performance,

6 wherein for a field-operator pair in an associated client DLL, the
7 rearranging comprises:
8 determining when a statement is processing intensive; and
9 rearranging statements of the policy to check at least some of the
10 processing intensive statements after less processing intensive statements.

1 17. The method of claim 16 wherein the optimization procedure further
2 comprises eliminating at least some repeated statements including repeated
3 portions of the statements, and
4 wherein as part of the compiling, the method comprises generating an
5 optimized policy transmission language with the eliminated statements and
6 eliminated repeated portions of the statements.

1 18. The method of claim 16 further comprising:
2 marking route attributes in a batch of routes as invariant for ones of the
3 route attributes that share values across differing route prefixes; and
4 caching results for statements associated with the marked attributes when
5 executing the compiled policy transmission language,
6 wherein upon continued execution of the compiled policy transmission
7 language, using the cached results for evaluations of subsequent statements in the
8 compiled policy transmission language which reference the marked attributes.

1 19. A policy compiler comprising:
2 means for generating a configuration data abstraction layer of a routing
3 policy, the configuration data abstraction layer to map configuration to an
4 intermediate layer comprising fields, operators and arguments; and
5 means for verifying the intermediate layer against a set of verification rules
6 for one or more client protocols including versions thereof, the policy repository
7 to generate compiled policy transmission language for use by the one or more
8 client protocols including versions thereof,
9 wherein the means for verifying is to verify the intermediate layer against
10 the set of verification rules for a particular one or more of a plurality of client

11 protocols including versions of the one or more of the plurality of client protocols,
12 and
13 wherein the means for verifying is to generate the compiled policy
14 transmission language for use by the one or more client protocols including
15 versions thereof.

1 20. A machine-readable medium that provides instructions, which when
2 executed by one or more processors, cause the processors to perform operations
3 comprising generating compiled policy for execution by one or more client
4 protocols including one or more versions of the client protocols,
5 wherein the instructions, when further executed by one or more of the
6 processors cause the processors to perform operations further comprising:
7 generating a configuration data abstraction layer for a routing policy to
8 map configuration to an intermediate layer;
9 verifying the intermediate layer against a set of rules for the one or more
10 client protocols including the one or more versions thereof; and
11 generating compiled policy transmission language when statements of the
12 intermediate layer are verified against the set of rules for the one or more client
13 protocols including the one or more versions thereof.

1 21. A method for optimizing a compiled policy, the method comprising:
2 marking statements in a compiled policy as invariant for ones of the route
3 attributes that share values across differing route prefixes; and
4 caching results for statements associated with the marked attributes when
5 executing the compiled policy,
6 wherein upon continued execution of the compiled policy, using the
7 cached results for evaluations of subsequent statements in the policy which
8 reference the marked attributes.

1 22. The method of claim 21 wherein marking comprises statements in a
2 compiled representation of the policy as invariant when one or more attributes do
3 not change with respect to received routes that are to be executed with respect to
4 an associated policy.

1 23. The method of claim 22 wherein caching comprises temporarily
2 storing results of executing statements associated with the marked attributes when
3 executing the compiled policy, and
4 wherein using comprises using the cached results for the marked attributes
5 to obtain a result for a route, the result being one of a modified route, an accepted
6 and unchanged route, or dropped route.

1 24. The method of claim 23 further comprising:
2 prior to the marking, receiving a route update message from a peer, the
3 route update message being associated one of a plurality of client protocols and
4 associated with a version of one of the client protocols,
5 wherein the route update message comprises a plurality of updated routes
6 wherein at least some of the updated routes with differing prefixes have attributes
7 of identical values, and
8 wherein marking comprising marking the attributes of identical values of
9 the updated routes.

1 25. The method of claim 24 wherein the executing comprises running the
2 compiled policy on an execution engine associated with a client protocol to apply
3 the updated routes,
4 wherein execution of the compiled policy, including the use of the cached
5 results, results in reduced processing operations for the associated execution
6 engine.

1 26. The method of claim 25 wherein the caching comprising caching
2 results of comparison statements that reference route attributes marked as
3 invariant, and
4 wherein the executing comprises refraining from executing the comparison
5 statements in the compiled policy having cached results.

1 27. The method of claim 21 wherein the compiled policy is in a policy
2 transmission language, and
3 wherein the method further comprises, prior to the marking, generating the
4 compiled policy for execution by one or more client protocols or more than one
5 version the client protocols, the generating comprising:
6 generating a configuration data abstraction layer for a routing policy to
7 map configuration to an intermediate layer;
8 verifying the intermediate layer against a set of rules for the one or more
9 client protocols including the one or more versions thereof; and
10 generating compiled policy transmission language when statements of the
11 intermediate layer are verified against the set of rules for the one or more client
12 protocols including the one or more versions thereof.

1 28. A compiler comprising:
2 one or more processors to implement a plurality of client protocols
3 including versions thereof to mark statements in a route update message as
4 invariant for route attributes that share values across differing route prefixes; and
5 an execution engine associated with one of the client protocols including
6 versions thereof to cache results for statements associated with the marked
7 attributes when executing the compiled policy,
8 wherein upon continued execution of the compiled policy, the execution
9 engine is to use the cached results for evaluations of subsequent statements in the
10 policy which reference the marked attributes.

1 29. The compiler of claim 28 wherein one of the client protocols including
2 versions thereof is to mark route attributes in a compiled batch of routes as
3 invariant when one or more attributes do not change with respect to received
4 routes that are to be executed with respect to an associated routing policy.

1 30. The compiler of claim 29 wherein as part of caching, the execution
2 engine is to temporarily store execution results for statements associated with the
3 marked attributes when executing the compiled policy, and
4 wherein the execution engine is to use the cached results for the marked
5 attributes to obtain a result for a route, the result being one of a modified route, an
6 accepted and unchanged route, or dropped route.

1 31. The compiler of claim 30 wherein one of the client protocols including
2 versions thereof is to receive a route update message from a peer router, the route
3 update message being associated one of the client protocols and associated with
4 one of the versions of the client protocols,
5 wherein the route update message comprises a plurality of updated routes
6 wherein at least some of the updated routes with differing prefixes have attributes
7 of identical values, and
8 wherein the one of the client protocols is to further mark the attributes of
9 identical values.

1 32. The compiler of claim 31 wherein the execution engine is to run the
2 compiled policy using the received routes,
3 wherein when executing the compiled policy, the execution engine's use of
4 the cached results is to reduce processing operations for the execution engine.

1 33. The compiler of claim 32 wherein the execution engine is to cache
2 results of comparison statements that reference route attributes marked as
3 invariant, and is to refrain from executing the comparison statements in the
4 compiled policy having the cached results.

1 34. The compiler of claim 28 further comprising a policy repository and a
2 system controller, and
3 wherein the system controller is to generate a configuration data
4 abstraction layer of a routing policy, the configuration data abstraction layer to
5 map configuration to an intermediate layer comprising fields, operators and
6 arguments,
7 wherein the policy repository is to verify the intermediate layer against a
8 set of verification rules for one or more of the client protocols including versions
9 thereof, and
10 wherein the policy repository is to generate the compiled policy in the
11 policy transmission language for use by the one or more client protocols including
12 versions thereof.

1 35. A compiler comprising:
2 a plurality of means for marking route attributes in a route and statements
3 in a compiled policy as invariant for ones of the route attributes that share values
4 across differing route prefixes;
5 means, associated with the means for marking, to cache results for
6 statements associated with the marked attributes; and
7 means for executing the compiled policy with the marked route attributes,

8 wherein upon continued execution of the compiled policy, the means for
9 executing is to use the cached results for evaluations of subsequent statements in
10 the policy which reference the marked attributes.

1 36. A machine-readable medium that provides instructions, which when
2 executed by one or more processors, cause the processors to perform operations
3 comprising:

4 marking route attributes in a compiled policy as invariant for ones of the
5 route attributes that share values across differing route prefixes; and

6 caching results for statements associated with the marked attributes when
7 executing the compiled policy,

8 wherein upon continued execution of the compiled policy, using the
9 cached results for evaluations of subsequent statements in the policy which
10 reference the marked attributes,

11 wherein the marking comprising marking route attributes in a compiled
12 representation of the policy as invariant when one or more attributes do not
13 change with respect to received routes that are to be executed with respect to an
14 associated policy, and

15 wherein the caching comprises temporarily storing results of the executing
16 for statements associated with the marked attributes when executing the compiled
17 policy, and

18 wherein using comprises using the cached results for the marked attributes
19 to obtain a result for a route, the result being one of a modified route, an accepted
20 and unchanged route, or dropped route.

1 37. A method for running at least first and second versions of a client
2 protocol on a router, the method comprising:
3 associating a dynamic registration process with each of the at least first and
4 second versions of the client protocol,
5 wherein the dynamic registration processes are to provide a policy
6 repository with a location of a dynamic link library (DLL) for an associated one of
7 the versions of the client protocol,
8 wherein the dynamic registration processes are to further provide the
9 policy repository with a registration location for configuration space to be used by
10 a routing policy for use when registering with the associated one of the versions of
11 the client protocols, and
12 wherein each of the versions of the client protocol has a configuration
13 space and a DLL associated therewith.

1 38. The method of claim 37 wherein when a routing policy is to be
2 registered with the second version of the client protocol while the first version of
3 the client protocol is running the router, the method further comprises:
4 verifying policy statements of the routing policy with verification rules in
5 the DLL associated with the second version of the client protocol; and
6 using the configuration space associated with the second version of the
7 client protocol for the verification.

1 39. The method of claim 38 wherein first and second DLLs are associated
2 respectively with the first and second versions of the client protocol,
3 wherein the DLLs comprise verification rules for the associated version of
4 the client protocol, and
5 wherein the router is to support the addition of the second version of the
6 client protocol while the first version is running on the router.

1 40. The method of claim 39 further comprising associating an additional
2 dynamic registration process for each of at least one additional client protocol.

1 41. The method of claim 40 wherein the at least first and second versions
2 of the client protocol include at least two versions of a border gateway protocol
3 (BGP), and
4 wherein the at least one additional client protocol comprises at least one
5 of an open-shortest-path-first (OSPF) client protocol and an intermediate system
6 to intermediate system (IS-IS) client protocol.

1 42. The method of claim 41 wherein when the policy statements of the
2 routing policy are verified, the method comprises;
3 moving at least some peer routers to the second version of the client
4 protocol; and
5 removing the at least some peer routers from the first version of the client
6 protocol.

1 43. The method of claim 42 wherein the dynamic registration processes
2 comprise software agents operating within the router.

1 44. The method of claim 35 further comprising:
2 querying, by a parser, the policy repository for capabilities of configuration
3 associated with the second version of the client protocol when the second version
4 of the client protocol is being added.

1 45. A router comprising:
2 a policy repository; and
3 one or more processors to implement dynamic registration processes, each
4 of the processes associated with at least first and second versions of a client
5 protocol,
6 wherein the dynamic registration processes are to provide the policy
7 repository with a location of a dynamic link library (DLL) for an associated one of
8 the versions of the client protocol,
9 wherein the dynamic registration processes are to further provide the
10 policy repository with a registration location for configuration space to be used by
11 a routing policy for use when registering with the associated one of the versions of
12 the client protocols, and
13 wherein each of the versions of the client protocol has a configuration
14 space and a DLL associated therewith.

1 46. The router of claim 45 wherein when a routing policy is to be
2 registered with the second version of the client protocol while the first version of
3 the client protocol is running the router, the second version of the client protocol
4 is to verify policy statements of the routing policy with verification rules in the
5 DLL associated with the second version of the client protocol and use the
6 configuration space associated with the second version of the client protocol for
7 the verification.

1 47. The router of claim 46 wherein first and second DLLs are associated
2 respectively with the first and second versions of the client protocol,
3 wherein the DLLs comprise verification rules for the associated version of
4 the client protocol, and
5 wherein the router is to support the addition of the second version of the
6 client protocol while the first version is running on the router.

1 48. The router of claim 47 wherein the one or more processors are to
2 further implement additional dynamic registration processes associated for each of
3 at least one additional client protocol.

1 49. The router of claim 48 wherein the at least first and second versions of
2 the client protocol include at least two versions of a border gateway protocol
3 (BGP), and

4 wherein the at least one additional client protocol comprises at least one
5 of an open-shortest-path-first (OSPF) client protocol and an intermediate system
6 to intermediate system (IS-IS) client protocol.

1 50. The router of claim 49 wherein when the policy statements of the
2 routing policy are verified, the second version of the client protocol is to be
3 requested to move at least some peer routers to the second version of the client
4 protocol and is to be requested to remove the at least some peer routers from the
5 first version of the client protocol.

1 51. The router of claim 50 wherein the dynamic registration processes
2 comprise software agents operating on the one or more processors.

1 52. The router of claim 51 further comprising a parser to query the policy
2 repository for capabilities of configuration associated with the second version of
3 the client protocol when the second version of the client protocol is being added.

1 53. A router comprising:
2 means for coordinating routing policies; and
3 means for associating dynamic registration processes with at least first and
4 second versions of a client protocol,

5 wherein the dynamic registration processes are to provide means for
6 coordinating with a location of a dynamic link library (DLL) for an associated one
7 of the versions of the client protocol,

8 wherein the dynamic registration processes are to further provide the
9 means for coordinating with a registration location for configuration space to be

10 used by a routing policy for use when registering with the associated one of the
11 versions of the client protocols, and
12 wherein each of the versions of the client protocol has a configuration
13 space and a DLL associated therewith,
14 wherein when a routing policy is to be registered with the second version
15 of the client protocol while the first version of the client protocol is running the
16 router, the second version of the client protocol includes means for verifying
17 policy statements of the routing policy with verification rules in the DLL
18 associated with the second version of the client protocol, and means for using the
19 configuration space associated with the second version of the client protocol for
20 the verification.

1 54. A machine-readable medium that provides instructions, which when
2 executed by one or more processors, cause the processors to perform operations
3 for running at least first and second versions of a client protocol on a router, the
4 operations comprising:
5 associating a dynamic registration process with each of the at least first and
6 second versions of the client protocol,
7 wherein the dynamic registration processes are to provide a policy
8 repository with a location of a dynamic link library (DLL) for an associated one of
9 the versions of the client protocol,
10 wherein the dynamic registration processes are to further provide the
11 policy repository with a registration location for configuration space to be used by
12 a routing policy for use when registering with the associated one of the versions of
13 the client protocols, and
14 wherein each of the versions of the client protocol has a configuration
15 space and a DLL associated therewith.

1 55. A method of verifying statements of a routing policy prior to compiling
2 the routing policy, the method comprising:
3 generating libraries for attach points associated with one or more versions
4 of one or more client protocols, the libraries to include capabilities for the one or
5 more versions of the one or more client protocols; and
6 individually checking statements of a routing policy against the
7 capabilities of one or more of the attach points.

1 56. The method of claim 55 wherein the libraries describe fields supported
2 by an associated one of the versions of the one or more client protocols,
3 wherein the libraries further describe operators supported by the fields for
4 the associated one of the versions of the one or more client protocols, and
5 wherein the libraries include a verification function for field-operator
6 combinations associated with one of the versions of the one or more client
7 protocols.

1 57. The method of claim 56 further comprising individually checking the
2 statements when the routing policy is initially being defined.

1 58. The method of claim 57 further comprising informing a policy
2 repository to load one of the libraries associated with a particular one of the attach
3 points associated with one of the versions of the one or more client protocols,
4 wherein the informing is performed by a dynamic registration process
5 associated with one of the versions of the one or more client protocols.

1 59. The method of claim 58 wherein the versions of the one or more client
2 protocols have one or more attach points, the attach points comprising at least
3 some of incoming neighbor points, outgoing neighbor points, aggregation points
4 and dampening points.

1 60. The method of claim 58 further comprising:

2 during the checking, providing an indication when a statement of the
3 routing policy currently being checked is not supported by the capabilities of one
4 of the attach points; and
5 receiving a selection by a user, the selection to direct a policy repository to
6 either skip the statement not supported by an attach point, or reject the policy
7 when the statement is not supported by an attach point.

1 61. The method of claim 60 wherein the indication is selectable and
2 includes one of ignoring the statement when an operation is not supported by an
3 attach point, ignoring and warning that the operation is not supported by an attach
4 point, and rejecting the policy when the operation is not supported by an attach
5 point.

1 62. The method of claim 60 further comprising:
2 compiling the routing policy when either all statements are verified or
3 when statements not supported by an attach point are skipped; and
4 when executing the compiled routing policy by an execution engine of a
5 version of a client protocol, skipping the statements not supported by an
6 associated attach point,
7 wherein the skipped statements comprise comparison operators.

1 63. The method of claim 61 further comprising performing a verification
2 for statements of the policy having comparison operators when applying a
3 compiled policy to an attach point.

1 64. The method of claim 55 further comprising providing a notification
2 when all statements of a policy are verified for at least one attach point, the
3 notification to include which of the one or more attach points the policy is valid.

1 65. A policy compiler comprising:
2 a policy repository to store capabilities for attach points,
3 one or more processors to implement a plurality of dynamic registration
4 processes to inform the policy repository of the capabilities for the attach points,
5 the dynamic registration processes being associated with a version of a client
6 protocol; and
7 libraries for attach points associated with one or more versions of one or
8 more client protocols, the libraries to include capabilities for the one or more
9 versions of the one or more client protocols,
10 wherein the policy repository is to check statements of a routing policy
11 against the capabilities of one or more of the attach points during generation of the
12 policy.

1 66. The policy compiler of claim 65 wherein the libraries describe fields
2 supported by an associated one of the versions of the one or more client protocols,
3 wherein the libraries further describe operators supported by the fields for
4 the associated one of the versions of the one or more client protocols, and
5 wherein the libraries include a verification function for field-operator
6 combinations associated with one of the versions of the one or more client
7 protocols.

1 67. The policy compiler of claim 66 wherein the policy repository is to
2 further individually check statements when the routing policy is initially being
3 defined.

1 68. The policy compiler of claim 67 wherein one of the dynamic
2 registration processes associated with one of the versions of the one or more client
3 protocols is to inform a policy repository to load one of the libraries associated
4 with a particular one of the attach points associated with one of the versions of the
5 one or more client protocols.

1 69. The policy compiler of claim 68 wherein the versions of the one or
2 more client protocols have one or more attach points, the attach points comprising
3 at least some of incoming neighbor points, outgoing neighbor points, aggregation
4 points and dampening points.

1 70. The policy compiler of claim 68 further comprising an I/O, wherein
2 during the checking, the policy compiler is to provide an indication when a
3 statement of the routing policy currently being checked is not supported by the
4 capabilities of an attach point, and
5 wherein a selection by a user is to be received through the I/O, the
6 selection to direct the policy repository to either skip the statement not supported
7 by an attach point, or reject the policy when the statement is not supported by an
8 attach point.

1 71. The policy compiler of claim 70 wherein the indication is selectable
2 and includes one of ignoring the statement when an operation is not supported by
3 an attach point, ignoring warning that the operation is not supported by an attach
4 point, and rejecting the policy when the operation is not supported by an attach
5 point.

1 72. The policy compiler of claim 70 wherein the policy repository is to
2 compile the routing policy when either all statements are verified or when
3 statements not supported by an attach point are skipped, and
4 wherein the policy compiler further comprises an execution engine
5 associated with a version of a client protocol, the execution engine to execute the
6 compiled routing policy and to skip statements not supported by an associated
7 attach point.

1 73. A policy compiler comprising:
2 means for storing capabilities for attach points,
3 means for informing a policy repository of the capabilities for the attach
4 points, the means for informing being associated with a version of a client
5 protocol;

6 means for generating libraries for attach points associated with one or
7 more versions of one or more client protocols, the libraries to include capabilities
8 for the one or more versions of the one or more client protocols;
9 means for checking statements of a routing policy against the capabilities
10 of one or more of the attach points during generation of the policy;
11 means for providing an indication when a statement of a routing policy
12 currently being checked is not supported by the capabilities of one of the attach
13 points; and
14 means for receiving a selection by a user, the selection to direct the means
15 for checking to either skip the statement not supported by the attach point, or
16 reject the policy when the statement is not supported by the attach point.

1 74. A machine-readable medium that provides instructions, which when
2 executed by one or more processors, cause the processors to perform operations
3 for verifying statements of a routing policy, the operations comprising:
4 generating libraries for attach points associated with one or more versions
5 of one or more client protocols, the libraries to include capabilities for the one or
6 more versions of the one or more client protocols; and
7 individually checking statements of a routing policy against the
8 capabilities of one or more of the attach points.

1 75. A method for transitioning between routing policies, wherein a first
2 configuration state is associated with the first route policy, wherein a second
3 configuration state is associated with a second route policy, and wherein the first
4 configuration state is a current configuration state of a router, the method
5 comprising:
6 grouping configuration elements of the second route policy into policy
7 statements and sets;
8 verifying the grouped configuration elements against capabilities with
9 verification rules associated with one or more versions of one or more client
10 protocols;
11 compiling statements of the second route policy when verified for at least
12 one of the one or more versions of the one or more client protocols; and
13 notifying the at least one of the one or more versions of the one or more
14 client protocols that the second route policy is to take effect.

1 76. The method of claim 75 further comprising, after the compiling,
2 overwriting a compiled version of the first route policy with the compiled second
3 route policy, the second route policy being in a compiled policy transmission
4 language.

1 77. The method of claim 76 wherein in response to the notifying, the at
2 least one of the one or more versions of the one or more client protocols is to
3 execute the compiled second route policy, and
4 wherein the router is to apply the second route policy to attach points
5 associated with the at least one of the one or more versions of the one or more
6 client protocols.

1 78. The method of claim 77 wherein prior to and during the grouping, the
2 verifying, the compiling and the notifying, the method further comprising running
3 the first route policy.

1 79. The method of claim 75 further comprising receiving statements
2 representing the second route policy from an operator.

1 80. The method of claim 77 wherein the second route policy comprises
2 policy statements that identify fields, operators and arguments, and
3 wherein verifying comprises, for each policy statement:
4 verifying fields for the one or more versions of the one or more client
5 protocols;
6 verifying field-operator pairings for the one or more versions of the one or
7 more client protocols; and
8 verifying arguments for each field-operator pairing for the one or more
9 versions of the one or more client protocols.

1 81. A policy compiler to transition a router from a first configuration state
2 associated with the first route policy to a second configuration state associated
3 with a second route policy, the policy compiler comprising:
4 a policy repository to group configuration elements of the second route
5 policy into policy statements and sets, and to verify the grouped configuration
6 elements against capabilities with verification rules associated with one or more
7 versions of one or more client protocols, the policy repository to further compile
8 statements of the second route policy when verified for at least one of the one or
9 more versions of the one or more client protocols; and
10 a system controller to notify the at least one of the one or more versions of
11 the one or more client protocols that the second route policy is to take effect.

1 82. The policy compiler of claim 81 wherein after compiling, the policy
2 repository is to overwrite a compiled version of the first route policy with the
3 compiled second route policy, the second route policy being in a compiled policy
4 transmission language.

1 83. The policy compiler of claim 82 wherein the at least one of the one or
2 more versions of the one or more client protocols is to execute the compiled
3 second route policy, and

4 wherein the router is to apply the second route policy to attach points
5 associated with the at least one of the one or more versions of the one or more
6 client protocols.

1 84. The policy compiler of claim 83, wherein prior to and during the policy
2 repository grouping, verifying, and compiling, the router is to run the first route
3 policy.

1 85. The policy compiler of claim 84 further comprising an I/O to receive
2 the statements representing the second policy from an operator.

1 86. The policy compiler of claim 85 wherein the second route policy
2 comprises policy statements that identify fields, operators and arguments, and
3 wherein the policy repository is to verify, for route policy statement, fields
4 for the one or more versions of the one or more client protocols, field-operator
5 pairing for the one or more versions of one or more client protocols, and
6 arguments for each field-operator pairing for the one or more versions of the one
7 or more client protocols.

1 87. A policy compiler to transition a router from a first configuration state
2 associated with the first route policy to a second configuration state associated
3 with a second route policy, the policy compiler comprising:
4 means for grouping configuration elements of the second route policy into
5 policy statements and sets;
6 means for verifying the grouped configuration elements against
7 capabilities with verification rules associated with one or more versions of one or
8 more client protocols;
9 means for compiling statements of the second route policy when verified
10 for at least one of the one or more versions of the one or more client protocols;
11 and
12 means for notifying the at least one of the one or more versions of the one
13 or more client protocols that the second route policy is to take effect.

1 88. A machine-readable medium that provides instructions, which when
2 executed by one or more processors, cause the processors to perform operations
3 for transitioning between routing policies, wherein a first configuration state is
4 associated with the first route policy, wherein a second configuration state is
5 associated with a second route policy, and wherein the first configuration state is a
6 current configuration state of a router, the operations comprising:
7 grouping configuration elements of the second route policy into policy
8 statements and sets;
9 verifying the grouped configuration elements against capabilities with
10 verification rules associated with one or more versions of one or more client
11 protocols;
12 compiling statements of the second route policy when verified for at least
13 one of the one or more versions of the one or more client protocols; and
14 notifying the at least one of the one or more versions of the one or more
15 client protocols that the second route policy is to take effect.